UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte MARC TREMBLAY

Appeal 2007-1284
Application 09/583,097[1]
Technology Center 2100

Decided: July 9, 2007

Before ANITA PELLMAN GROSS, HOWARD B. BLANKENSHIP, and JOHN A. JEFFERY, *Administrative Patent Judges*.

JEFFERY, *Administrative Patent Judge*.

DECISION ON APPEAL

Appellant appeals under 35 U.S.C. § 134 from the Examiner's rejection of claims 9-34. Claims 1-8, 35, and 36 have been cancelled.[2] We have jurisdiction under 35 U.S.C. § 6(b). We affirm-in-part.

---

[1] This application is a divisional application of Application No. 08/662,582, filed June 11, 1996, now U.S. Pat. 5,958,042.

STATEMENT OF THE CASE

Appellant invented a central processing unit including a grouping logic circuit that determines simultaneously dispatchable instructions in a processor cycle. Specifically, the logic circuit includes a number of pipeline stages such that resource allocation and data dependency checks can be performed over multiple processor cycles. Such a feature enables dispatching a large number of instructions simultaneously.[3] Claim 9 is illustrative:

> 9. A superscalar processor that, for a given instruction instance, performs, over plural execution cycles of the superscalar processor, instruction grouping for dispatch, including both intra-group and inter-group dependency checking, wherein the instruction grouping for dispatch takes the plural execution cycles to complete.

The Examiner relies on the following prior art reference to show unpatentability:

Sites	US 5,193,167	Mar. 9, 1993

Claims 9-34 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Sites.

---

[2] Claims 35 and 36 were cancelled in an Amendment After Final Rejection filed June 9, 2005 which was entered by the Examiner. *See* Answer 2 and Reply Br. 2.

[3] *See generally* Specification 3:17-33.

Rather than repeat the arguments of Appellant or the Examiner, we refer to the Briefs and the Answer[4] for their respective details. In this decision, we have considered only those arguments actually made by Appellant. Arguments which Appellant could have made but did not make in the Briefs have not been considered and are deemed to be waived. *See* 37 C.F.R. § 41.37(c)(1)(vii).

OPINION

Anticipation is established only when a single prior art reference discloses, expressly or under the principles of inherency, each and every element of a claimed invention as well as disclosing structure which is capable of performing the recited functional limitations. *RCA Corp. v. Applied Digital Data Systems, Inc.*, 730 F.2d 1440, 1444, 221 USPQ 385, 388 (Fed. Cir. 1984); *W.L. Gore and Associates, Inc. v. Garlock, Inc.*, 721 F.2d 1540, 1554, 220 USPQ 303, 313 (Fed. Cir. 1983).

The Examiner has indicated how the claimed invention is deemed to be fully met by the disclosure of Sites (Answer 3-15). Regarding independent claim 26, Appellant argues that Sites fails to disclose dependency checking during plural pipelined execution cycles as claimed. Appellant contends that any dependency checking in Sites occurs during a single cycle, specifically in the fourth stage (S3) of Sites' pipeline.

---

[4] A first Examiner's Answer was mailed on Sept. 22, 2005. In response to a subsequent order from the Board noting various deficiencies in the record, a second Examiner's Answer was mailed on Nov. 17, 2006. We refer to the Nov. 2006 Answer throughout this opinion.

According to Appellant, dependency checking in Sites necessarily employs information regarding operations and register addresses to identify source and target conflicts between instructions. Because such information is obtained after register addresses and operations are decoded (in stage S2), any dependency checking in Sites must therefore occur after the decoding stage S2 (i.e., in stage S3) (Br. 7-9).

The Examiner argues that two types of dependency checks occur in different execution cycles in Sites' system. The first type, which the Examiner characterizes as "intra-group dependency checking," occurs in swap stage S1. The Examiner notes that in this stage, two fetched instructions are evaluated to determine whether they can be issued simultaneously -- an evaluation based on the specific type of instruction for each instruction comprising the fetched pair. That is, if the two types are compatible, then the pair of instructions will issue simultaneously (i.e., dual issue will occur). This process, according to the Examiner, constitutes "dependency checking" since the decision to issue a given instruction in a pair effectively depends on the other instruction's type (Answer 16-17).

The second type of dependency checking, which the Examiner characterizes as "inter-group dependency checking," occurs after the decoding stage S2. The Examiner notes that, after decoding, the system determines whether resources are available for both instructions to issue. According to the Examiner, such checking for available resources involves checking if a previously-issued group is using resources that the current group requires (Answer 17-18).

Appellant responds that Sites' swap stage S1 does not constitute intra-group dependency checking. According to Appellant, the inability to issue one instruction with another is not based on any dependency between instructions, but rather based on whether two instructions can be executed by the execution unit to which they are slotted (Reply Br. 4-5). Appellant further contends that Sites' group of two instructions is not a "dispatch group" as claimed (i.e., a group of instructions that may be dispatched together under certain conditions), but rather a "fetch group" (Reply Br. 5-6).

We will sustain the Examiner's rejection of independent claim 26 essentially for the reasons stated by the Examiner.[5] Sites' instruction unit 18 decodes two instructions in parallel in decoders 23, 24, and then checks that the required resources are available for instructions by check circuitry 25. If (1) resources are available, *and* (2) dual issue is possible, then both instructions are issued. However, if (1) resources are available for only the first instruction, *or* (2) the instructions cannot be dual issued, then only the first instruction is issued (Sites, col. 6, ll. 18-34; Fig. 3) (emphasis added).

---

[5] Appellant's request that prosecution be reopened for an alleged new ground of rejection on Pages 3 and 4 of the Reply Brief is a petitionable matter under 37 CFR 1.181 – not an appealable matter. *See* MPEP § 1002.02(c) (noting that petitions or requests to reopen prosecution are decided by Technology Center Directors). Because we do not have jurisdiction over this matter, it is therefore not before us. *See* MPEP § 706.01 ("[T]he Board will not hear or decide issues pertaining to objections and formal matters which are not properly before the Board."); *see also* MPEP § 1201 ("The Board will not ordinarily hear a question that should be decided by the Director on petition....").

Specifically, the first four stages of a seven stage pipeline are executed in the instruction unit 18. These stages are discussed in column 10, lines 15-47 of Sites and are summarized as follows:

Stage	Description	Function
S0	Instruction fetch stage	Instruction unit fetches two new instructions from the instruction cache
S1	Swap stage	Both fetched instructions are evaluated to see if they can be issued simultaneously
S2	Decode stage	Both instructions are decoded in decoders 23, 24 to produce control signals 28, 29 and register addresses 26, 27
S3	Register file access stage	(1) Register file access stage (for operate instructions) (2) Issue check decision point (for all instructions) (3) Instruction issue stage

Circuitry 25 makes the dual issue decision in accordance with a predetermined requirement in which only certain types of instructions can issue together. That is, only one instruction from the list of instructions in Column A of Sites can issue with only one instruction from Column B. Circuitry 25 also determines if resources are available before allowing two instructions to issue in the same cycle (Sites, col. 9, ll. 45-63; Fig. 3).

Based on this functionality, we agree with the Examiner that "intra-group dependency checking" occurs in swap stage S1 of Sites, and "inter-group dependency checking" occurs after decoding (i.e., stage S3). As indicated above, both fetched instructions are evaluated in swap stage S1 to

see if they can be issued simultaneously. This evaluation, in our view, most reasonably corresponds to the "dual issue" decision noted above. Moreover, the resource availability determination -- a determination independent of the dual issue decision -- reasonably corresponds to the decision that is made in stage S3 as indicated by its issue check functionality noted above.

Significantly, Figure 3 of Sites shows two distinct "Issue Check" boxes as part of instruction unit 18: (1) an "Issue Check" box before decoders 23, 24, and (2) an "Issue Check" box after the decoders. The clear import of this teaching, taken with Sites' two sequential determinations made by the instruction unit noted above (i.e., (1) dual issue and (2) resource availability determinations), is that the dual issue determination corresponds to the first "Issue Check" box, and the resource availability determination corresponds to the second "Issue Check" box.

We find unpersuasive Appellant's contention that since circuitry 25 performs the dual issue determination and is shown in Fig. 3 as following the decode stage S2, the dual issue decision therefore follows decoding. We recognize that Sites teaches that circuitry 25 performs both the dual issue and resource availability determinations. *See* Sites, col. 9, lines 45 and 61-63. Also, we acknowledge that numeral 25 is directed to only one "Issue Check" box in Figure 3 (i.e., the box after the decoders) and that the preceding "Issue Check" box (i.e., the box before the decoders) has no associated reference numeral.

This depiction, however, does not mean that both determinations must necessarily follow decoding as Appellant suggests. Although it is unclear why the first "Issue Check" box in Figure 3 lacks a corresponding reference numeral, what is clear from the figure is that an "Issue Check" occurs before

and after decoding. Furthermore, as we noted previously, Sites teaches that independent evaluations occur before decoding (i.e., in swap stage S1) and after decoding (in stage S3). The fact that numeral 25 is directed to only one "Issue Check" box in Figure 3 (i.e., after the decoders) does not negate the clear import of the reference's teachings taken as a whole. Based on the overall teachings of Sites, we presume that circuitry 25 is associated with the first and second "Issue Checks" – an interpretation acknowledged by the Examiner.[6] For at least these reasons, we agree with the Examiner that Sites discloses performing dependency checks during plural pipelined execution cycles as claimed.

In addition, we are not persuaded by Appellant's contention that Sites fails to disclose a "dispatch group" of instructions (Reply Br. 5-6). Although two instructions are fetched in stage S0 of Sites, these fetched instructions are then evaluated in swap stage S1 for issuance or dispatch. Once fetched, the group of instructions is effectively converted to a "dispatch group."

For the foregoing reasons, we will sustain the Examiner's rejection of independent claim 26. Since Appellant has not separately argued the patentability of dependent claims 27 and 29, these claims fall with independent claim 26. *See In re Nielson*, 816 F.2d 1567, 1572, 2 USPQ2d 1525, 1528 (Fed. Cir. 1987); *see also* 37 C.F.R. § 41.37(c)(1)(vii).

---

[6] *See* Answer 24 ("The first issue check logic [in Figure 3] does not have a reference number associated with it. It may very well be a part of circuitry 25.").

We will also sustain the Examiner's rejection of claim 28. As the Examiner indicates, all that the claim requires is that intra-group dependency checking and within-group resource allocation occur in successive processor cycles (Answer 21). As we indicated previously, intra-group dependency checking in Sites occurs in swap stage S1. We also agree with the Examiner that resource allocation for instructions in the group occurs after decoding – a stage that follows the intra-group dependency checks. Claim 28 is therefore fully met by Sites. Appellant's arguments to the contrary on pages 9-11 of the Brief are simply not commensurate with the scope of the claim.

We will not, however, sustain the Examiner's rejection of claim 31. The claim recites, in pertinent part, initiating inter-group dependency checking in a cycle of processor execution *prior to* the non-deterministic dependency condition checking.

The Examiner contends that Sites' inter-group dependency checking fully meets evaluating "non-deterministic" conditions given the broadest reasonable interpretation of "non-deterministic."[7] The Examiner also contends that before this non-deterministic checking occurs, it must be initiated (Answer 13).

Based on the broadest reasonable interpretation of the term "non-deterministic dependency checking" when read in light of Appellant's disclosure, we decline to adopt the Examiner's construction of the term as corresponding to inter-group dependency checking. For this reason alone, we cannot sustain the Examiner's rejection. But even with the Examiner's

[7] The Examiner notes that "non-deterministic" is a form of the word "nondeterminism" which is defined as "a property of a computation having more than one result" (Answer 13).

construction, we fail to see how the claim is fully met by Sites. The Examiner's interpretation, in effect, results in the same method steps occurring in succession – a result that simply strains any reasonable construction of the claimed limitations.

The Examiner appears to rely primarily on the limitation calling for "inter-group dependency checking" to be *initiated* to establish that the step occurs prior to the non-deterministic dependency condition checking step – a step which the Examiner also contends is inter-group dependency checking.[8] We do not agree with this reasoning. Claim differentiation principles and the limitations read in light of Appellant's disclosure strongly suggest that the recited steps of "checking non-deterministic conditions" and "inter-group dependency checking" correspond to distinct processes.

For at least these reasons, we will not sustain the Examiner's rejection of independent claim 31. Likewise, we will not sustain the Examiner's rejection of dependent claims 32-34. For similar reasons, we will also not sustain the Examiner's rejections of claims 16 and 30 which recite commensurate limitations.

We will, however, sustain the Examiner's rejection of independent claim 9. Although Appellant reiterates that any dependency checking in Sites is performed, if at all, in a single cycle after decode (i.e., in stage S3) (Br. 13-14), we find this argument unpersuasive for the reasons previously discussed.[9] The Examiner's rejection is therefore sustained. Since

---

[8] *See* Answer 13 ("[B]efore the non-deterministic dependency checking occurs (which is the inter-group checking), the inter-group checking must be initiated.").

[9] *See* pp. 5-8, *supra*, of this opinion.

Appellant has not separately argued the patentability of dependent claims 10-12 and 14, these claims fall with independent claim 9. *See* 37 C.F.R. § 41.37(c)(1)(vii).

We will also sustain the Examiner's rejection of claim 13. We find the Examiner's interpretation of "intra-group dependency checking" as including both the fetch stage S0 and the swap stage S1 reasonable. As the Examiner indicates, intra-group checks in the swap stage S1 would not be possible without first fetching the instructions (Answer 26-27). Since the intra-group dependency checking process spans these two stages (S0 and S1), the claim is fully met. Accordingly, we will sustain the Examiner's rejection of claim 13.

Regarding claim 15, Appellant argues that while Sites executes instructions based on a predicted taken (or untaken) branch, any data dependency checking and/or resource allocation checks are based on an *actual, then-current state* of Sites' processor, not a *predicted subsequent state* of the processor (Br. 14-15; Reply Br. 7) (emphasis added). The Examiner argues that the branch prediction capability of Sites fully meets a "predicted subsequent state" when the branch that is actually taken is the branch that was predicted (Answer 27-28).

We will sustain the Examiner's rejection of claim 15. Sites' branch prediction circuit 30 is used to predict branch addresses and cause address generating circuitry 32 to prefetch the instruction stream before needed (Sites, col. 6, ll. 42-48). We agree with the Examiner that if the branch that is taken is the predicted branch, then such a predicted-taken branch would be a "predicted subsequent state" of the processor giving the term its broadest reasonable interpretation. In such a case, Sites' data dependency and

resource allocation checks performed by instruction unit 18 would, at least in part, be based on the "predicted subsequent state" of the processor (i.e., the pre-fetched instruction stream for the predicted-taken branch). For at these reasons, the Examiner's rejection of claim 15 is therefore sustained.

We will not, however, sustain the Examiner's rejection of independent claim 17. We agree with Appellant that Sites does not reasonably disclose (1) computing over plural cycles a future state of the processor based on a prior state of the processor, and (2) selecting a group of instructions based thereon for dispatch as claimed. In our view, operations performed in Stages S0 through the end of Stage S3 are not reasonably interpreted as computing a future state of the processor based on a prior state of the processor as claimed.

For at least this reason, we will not sustain the Examiner's rejection of independent claim 17. Likewise, we will not sustain the Examiner's rejection of dependent claims 18-25.

DECISION

We have sustained the Examiner's rejection with respect to claims 9-15 and 26-29. We have not, however, sustained the Examiner's rejection with respect to claims 16-25 and 30-34. Therefore, the Examiner's decision rejecting claims 9-34 is affirmed-in-part.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED-IN-PART

pgc

Sun Microsystems, Inc.
c/o DARBY & DARBY P.C.
P.O. BOX 770
Church Street Station
NEW YORK NY 10008-0770